멀티미디어 서비스를 위한 리눅스 클러스터 파일 시스템의 접근 성능 개선*

홍재연**, 김형식
충남대학교 컴퓨터과학과
{lh0701, hskim}@cs.cnu.ac.kr

Improving Access Performance of the Linux Cluster File System for Multimedia Service

Jae-Youn Hong, Hyong-Shik Kim
Dept. of Computer Science, Chungnam National University

요 약

클러스터 구조는 고가용성(high availability)과 결함내성(fault tolerance)을 만족하고 확장성이 뛰어나기 때문에 클러스터 파일 시스템은 멀티미디어 서비스에 적합하다. 사용자 수준 클러스터 파일 시스템[1,2]은 멀티미디어 서비스에 특화된 기능을 제공하고 저장된 위치에 관계없이 파일이나 디렉토리에 접근할 수 있는 단일 시스템 이미지(single system image) 기술을 제공하지만 실제 저장된 위치에 따라 접근 시간의 편차가 발생한다.

본 논문에서는 메타 데이터 캐쉬와 시스템 버퍼를 이용한 사용자 수준 클러스터 파일 시스템의 성능 개선 방법을 제안하고 각각에 대하여 성능 개선 정도를 분석한다. 메타 데이터 캐쉬는 자주 참조되는 원격 노드의 메타 정보를 로컬 저장구조에 저장하고 시스템 버퍼는 데이터 블록의 쓰기 성능을 개선할 뿐만 아니라 선반입을 통하여 읽기 성능을 개선할 수 있다.

1. 서 론

클러스터 파일 시스템은 저장 용량 및 처리 용량에 대하여 확장성을 보장하고 장애가 발생하더라도 지속적인 서비스를 제공할 수 있을 뿐만 아니라 비용 대비 성능 비율에서 우수하기 때문에 멀티미디어 서비스 등의 특정 영역에 적합하다고 알려져 있다[3, 4]. 대부분의 클러스터 파일 시스템은 기존의 범용 파일 시스템을 각 노드에서 운용하면서, 단일 시스템 이미지(single system image)를 제공하기 위한 기법을 이용하여 단일 파일 시스템처럼 보이도록 구성된다.

이때 단일 시스템 이미지 기술은 하드웨어 수준, 미들웨어 수준, 혹은 응용 프로그램 수준에서 파일 시스템에 대한 접근 요구를 실제 저장된 위치와 상관없이 일관된 방식으로 서비스하는데 핵심적인 역할을 한다. 그러나 실제 저장된 위치에 따라 접근 시간의 편차가 발생할 수 있게 되므로 저장된 노드와 상관없이 빠른 접근 속도를 보장하는 것이 중요한 문제 중의 하나로 대두되었다.

클러스터 파일 시스템의 접근 시간은 파일 혹은 디렉토리에 관한 메타 정보를 획득하는 시간과 실제 데이터 블록을 획득하는 시간으로 구성된다. 본 논문에서는 클러스터 파일 시스템에서의 메타 정보에 대한 접근 시간을 단축하기 위해 [2]에서 제안된 메타 정보 캐쉬 기법이 성능 향상에 미치는 영향을 분석한다. 또한, 원격 노드의 데이터 블록 획득 시 발생하는 지연으로 인한 성능 저하를 방지하기 위해 시스템 버퍼를 설계하고 이를 통한 성능 향상의 정도를 측정하였다.

2. 문제정의

2.1 클러스터 파일 시스템

클러스터 파일 시스템에서는 파일의 입출력 시간을 단축하고 결함 허용 기능을 제공하기 위해서 단일 파일을 다중 노드에 분산하여 저장한다. 또한, 분산되어 있는 파일을 관리하기 용이하도록 기존의 파일 시스템과 동일한 디렉토리 구조를 제공해야 한다. 이때, 파일과 디렉토리를 위한 메타 정보는 클러스터 파일 시스템을 구성하는 각각의 노드에 중복되지 않도록 분산 저장된다.

예를 들면, 그림 1과 같이 클러스터 파일 시스템이 두 개의 노드로 구성될 경우 파일과 디렉토리에 대한 메타정보가 f-node와 d-node의 형태로 두 노드에 분산 저장된다. 파일에 대한 메타 정보는 'file3'의 저장 형태와 같이 부모 디렉토리와 다른 노드에 저장될 수 있다. 'file3'의 경우 노드 0에서는 파일 자체에 대한 정보를 갖고, 노드 1에서는 해당 파일이 어느 노드의 어느 위치에 존재하는지에 대한 정보를 'dir1'에 저장한다.

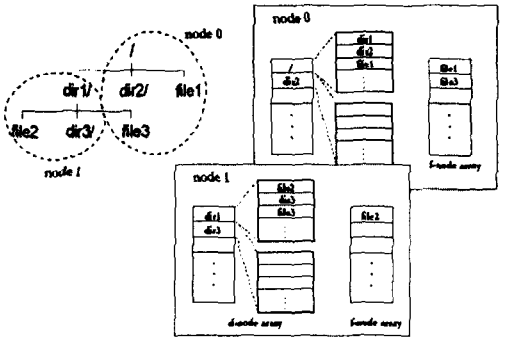

그림 1: 클러스터 파일 시스템의 메타 정보 저장 구조

한 파일의 실제 데이터 블록은 일정 크기로 분할되어 내부 표현용 이름으로 저장된다. 분할된 파일은 여러 노드에 분산되어 저장되는데,

* 이 논문은 정보통신부 지원 선도기반기술개발사업에 의하여 수행된 과제의 결과임.
** BK21 충남대학교 정보통신인력양성사업단의 지원을 받았음.

특정 노드를 지정하여 파일을 분산시킬 수 있다. 예를 들어, 클러스터 파일 시스템이 다섯 개의 노드로 구성되고 노드 1, 2, 4를 이용하여 파일을 분산시키고자 한다면 분할된 데이터 블록은 노드 1, 2, 4에 라운드 로빈 방식으로 저장될 것이다.

2.2 메타 데이터 캐쉬

앞에서 살펴본 바와 같이 클러스터 파일 시스템에서는 단일 파일 시스템을 위한 메타 정보를 다중 노드에 분산하여 저장한다. 그러나 단일 시스템 이미지를 제공하기 위해서는 메타 데이터의 위치와 상관없이 접근할 수 있어야 한다. 이때 원격 노드의 메타 정보를 획득하기 위한 지연 시간은 상당히 크다.

클러스터 파일 시스템에서는 메타 정보에 접근할 때 시간적 지역성의 특성이 발생함을 착안하여 각 노드에 메타 데이터 캐쉬를 구현하여 자주 참조되는 메타 정보를 저장함으로써 메타 데이터를 획득하기 위해 소요되는 시간을 단축시킨다.

쓰기 연산을 수행할 경우는 메타 정보를 변경시키므로 각 노드의 메타 데이터 캐쉬 사이에 불일치가 발생할 수 있다. 또한, 쓰기 연산은 데이터 블록을 변경하므로 동시에 두 개 이상 수행되거나 읽기 연산과 동시에 수행되지 않도록 보장해야 한다. 이를 위해 멀티미디어 데이터 서비스의 주요 기능인 읽기 연산의 성능을 저해하지 않으면서 일관성에 관한 문제를 해결하기 위한 알고리즘을 개발하여 적용하였다[2].

2.3 시스템 버퍼

단편적인 쓰기 연산에 대하여 매번 파일 시스템에 접근함으로써 발생하는 지연 시간을 단축하기 위해 시스템 버퍼라는 내부적인 데이터 버퍼 풀(pool)을 유지하여 직접 파일 시스템에 기록하지 않고 블록 단위로 쓰기 연산을 수행함으로써 디스크 접근 횟수를 감소시킨다.

이를 확장하여 미래에 읽기 연산을 위하여 참조될 데이터 블록을 미리 시스템 버퍼에 위치시키는 선반입 기법을 제공함으로써 데이터 블록을 획득하는 시간을 단축하여 읽기 연산의 성능을 향상시킨다. 선반입할 데이터 블록에 대한 정보는 응용프로그램 수준에서 제공된다고 가정한다.

3. 메타 데이터 캐쉬의 접근 성능 개선

메타 데이터 캐쉬를 사용하여 자주 참조되는 메타 데이터 중 일부를 로컬 메모리에 유지함으로써 원격 노드에 저장된 메타 정보에 접근하는 시간을 단축할 수 있다.

3.1 읽기 목적 open과 close의 성능 분석

그림 2는 읽기 목적 open과 close를 통해 메타 데이터 접근 시간을 측정한 것이다.

첫 번째와 네 번째 데이터는 메타 정보가 로컬 메모리와 원격 노드의 메모리에 존재하는 경우에 소요되는 시간이다. 그래프에서 볼 수 있듯이 로컬 메모리에 비해 원격 노드의 메타 정보에 접근하는데 상당한 시간이 더 소요됨을 알 수 있다.

두 번째 데이터는 원격 노드에 저장된 메타 정보를 캐쉬에서 획득할 경우의 성능을 측정한 것이다. 이 경우는 원격 노드에 접근할 필요가 없으므로 가장 이상적인 경우로 약 50배 정도의 성능 향상을 보인다.

세 번째 데이터는 메타 정보를 캐쉬에서 발견할 수 없을 경우의 성능을 나타낸다. 이 경우는 원격 노드로부터 메타 정보를 획득하여 캐쉬에 삽입한 후 캐쉬의 메타 정보를 이용하여 *open*을 수행하게 된다. 그러나, *close*을 수행할 때는 원격노드에 접근할 필요가 없으므로 메타 데이터 캐쉬를 사용하지 않는 경우보다 25% 정도의 성능 향상을 보인다.

3.2 쓰기 목적 open과 close의 성능 분석

쓰기 연산을 수행할 때는 메타 정보에 대한 불일치 문제가 발생할 수 있으므로 이를 해결하기 위해 일관성 유지 기법을 사용한다[2]. 이 기법은 모든 노드에서 캐쉬의 메타 정보가 무효화(invalidation)되도록 보장해야 하므로 패널티를 발생시킨다.

그림 3의 두 번째 데이터는 메타 데이터 캐쉬를 사용할 경우의 쓰기 목적 open과 close를 수행하기 위해 소요된 시간을 측정한 것이다. 일관성 유지 기법으로 인해 메타 데이터 캐쉬를 사용하지 않았을 경우보다 더 많은 시간이 소요됨을 알 수 있다. 그러나 멀티미디어 서비스에서는 쓰기 연산의 비중이 작다는 점을 고려하면 그림 3에서의 패널티가 전체 성능에 미치는 영향은 크지 않을 것으로 판단된다.

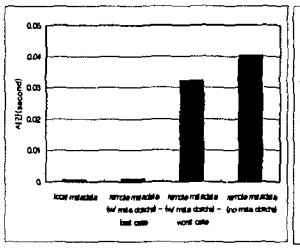

그림 2: 읽기 목적 open과 close의 성능 분석

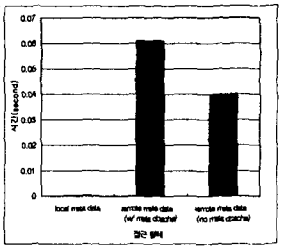

그림 3: 쓰기 목적 open과 close의 성능 분석

3.3 확장성 분석

다음의 그래프는 클러스터를 구성하는 노드 수의 증가가 일관성 유지 기법으로 인해 발생하는 패널티에 미치는 영향을 분석한 것이다.

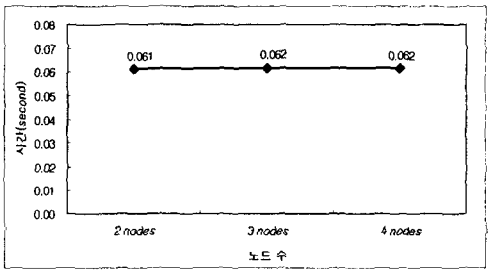

그림 4: 노드 수의 증가가 성능에 미치는 영향 분석

그림을 통해 노드 수의 증가가 패널티에 거의 영향을 미치지 않으므로 확장성이 있음을 알 수 있다.

3.4 Zipf 분포에 따른 메타 정보 참조 시 성능 분석

멀티미디어 서비스에서 대부분의 사용자는 인기 있는 몇 편의 영화만을 시청한다고 알려져 있다. 이와 같은 편중된 시청 경향을 Zipf 분포[5]를 이용하여 모델링 할 수 있다.

메타 정보에 대한 참조가 Zipf 분포를 따르고 메타 데이터 캐쉬의 크기가 167이며 총 참조 횟수가 4058일 때 읽기 목적 *open*과 *close*을 수행하는데 소요되는 평균 접근 시간은 그림 5과 같다. 참고로 이때의 적중률(*hit ratio*)은 58.9%이다.

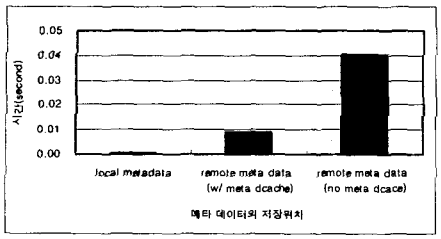

그림 5: Zipf 분포에 따른 성능 측정

4. 데이터 블럭의 접근 성능 개선

시스템 버퍼는 쓰기 연산을 위한 버퍼 공간을 제공함으로써 빈번한 쓰기 연산의 수행 시간을 단축하고 선반입 기법을 통해 읽기 연산의 성능을 개선한다.

4.1 읽기 연산에 대한 성능 분석

그림 6은 4K의 데이터 블록을 획득하기 위한 읽기 연산의 수행 시간을 측정한 것이다. 이때 시스템 버퍼의 크기는 16M이고 시스템 버퍼의 입출력을 위한 블록 크기는 16K이다. 이후의 모든 성능 측정에서 시스템 버퍼는 동일하게 구성되는 것으로 가정한다.

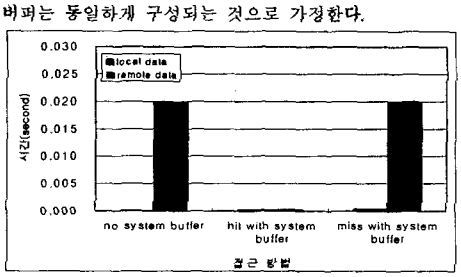

그림 6: 읽기 연산의 성능 측정

원격 노드에 존재하는 데이터 블록을 시스템 버퍼로부터 획득하는 경우는 로컬 디스크로부터 데이터 블록을 획득하는 경우와 비슷한 성능을 보인다. 시스템 버퍼에서 실패가 발생한 경우는 시스템 버퍼를 사용하지 않았을 경우와 비슷한 성능을 보이므로 시스템 버퍼를 유지하기 위한 패널티가 크지 않음을 알 수 있다.

4.2 고정된 크기, 순차적 읽기 접근에 대한 선반입 성능 분석

다음의 그래프는 선반입 기법이 고정된 크기로 순차 접근을 하는 읽기 연산에 미치는 영향을 측정한 것이다. 데이터 파일의 크기는 40M이고 4K씩 읽기 연산을 수행하였을 경우의 성능을 측정하였다. 그래프에는 생략되었지만 시스템 버퍼를 사용하지 않고 직접 파일 시스템에 접근할 경우는 104초의 시간이 소요되었다.

순차적인 접근을 하기 때문에 시스템 버퍼의 적중률이 높아 선반입 기법을 사용하지 않았을 경우에도 약 28초로써 시스템 버퍼를 사용하지 않았을 경우보다 4배 정도의 성능 향상을 보였다.

그림 7: 고정된 크기, 순차적인 접근에 대한 읽기 성능 측정

그림 7에서 distance는 선반입을 수행할 위치를 나타내며 현재 offset으로부터의 상대적인 위치를 블록 크기로 정규화한 것이다. distance가 너무 작을 경우는 선반입에 대한 요청이 완료되기 전에 읽기 요청이 발생하게 되므로 적당히 큰 distance를 사용할 때 최적의 성능을 얻을 수 있다. 제기된 결과는 distance가 10이상만 되면 최적의 성능에 도달함을 보인다.

4.3 임의의 크기, 임의의 접근에 대한 선반입 성능 분석

그림 8은 읽기 연산의 크기가 정규분포를 따르고 그림 7에서와 같이 평균이 4K일 때 40M의 데이터 파일에 임의의 접근을 하는 읽기 연산에 대한 선반입 성능을 측정한 것이다. 그래프에는 생략되었지만 시스템 버퍼를 사용하지 않았을 경우에는 순차적으로 접근할 경우와 비슷하게 100초 정도가 소요된다. 선반입 기법을 사용하였을 경우에는 distance가 1일 때 20% 정도의 시간을 단축함으로써 가장 높은 성능 향상을 얻을 수 있었고 distance가 커질 경우에는 성능 향상을 기대할 수 없었다. 제시된 결과는 본 논문에서 제안된 선반입 기법이 임의의 크기를 갖는 블록에 대한 임의의 접근에 대하여 부정적인 효과를 발생시키지 않으며 접근 형태에 대한 예측이 선반입 기법에 중요한 영향을 미침을 보인다.

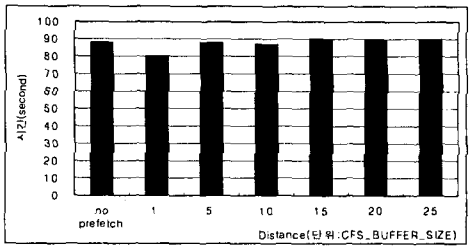

그림 8: 임의의 크기, 임의의 접근에 대한 읽기 성능 측정

5. 결론

본 논문에서는 클러스터 파일 시스템에서 멀티미디어 서비스를 수행할 때 발생하는 지연 문제를 해결하기 위한 방법으로 제안된 메타 데이터 캐쉬와 시스템 버퍼의 성능을 측정하였다.

읽기 목적 연산을 수행할 때 메타 데이터 캐쉬를 통해 원격 노드의 메타 정보에 접근하는 경우는 로컬 저장 구조로부터 정보를 획득하는 수준의 성능 향상을 얻을 수 있었다. 시스템 버퍼는 쓰기 연산을 위한 버퍼 공간을 제공함으로써 빈번한 쓰기 연산에 대한 접근 시간을 단축할 뿐만 아니라 선반입 기법을 통해 읽기 연산에 대한 접근 시간도 최고 1/100 정도로 단축하였다.

참고 문헌

[1] 강미연, 홍재연, 김형식, "효율적인 멀티미디어 서비스를 위한 리눅스 클러스터 파일 시스템", 한국정보과학회 2002년 춘계학술발표회, pp. 652-6, 2002년 4월.

[2] 홍재연, 김형식, "리눅스 클러스터 파일 시스템을 위한 메타 정보 캐쉬 기법", 한국정보과학회 2002년 추계학술발표회, pp. 316-318, 2002년 10월.

[3] R.Buyya, *High Performance Cluster Computing, Volume 1: Architectures and Systems*, Prentice-Hall, 1999.

[4] G.F. Pfister, *In Search of Clusters*, Prentice-Hall, 1998.

[5] Asit Dan, Dinkar Sitaram, Perwez Shahabuddin, *Dynamic batching policies for an on-demand video server*, Multimedia Systems, 4(3):112-21, Jun.1996.